子数组的最大乘积

给定一个长度为 N 的整数数组，只允许用乘法，不能用除法，计算任意（$N-1$）个数的组合乘积中最大的一组，并写出算法的时间复杂度。

我们把所有可能的（$N-1$）个数的组合找出来，分别计算它们的乘积，并比较大小。由于总共有 N 个（$N-1$）个数的组合，总的时间复杂度为 O（N^2），但显然这不是最好的解法。

分析与解法

【解法一】

在计算机科学中，时间和空间往往是一对矛盾体，不过，这里有一个优化的折中方法。可以通过“空间换时间”或“时间换空间”的策略来达到优化某一方面的效果。在这里，是否可以通过“空间换时间”来降低时间复杂度呢？

计算（$N-1$）个数的组合乘积，假设第 i 个（$0 \le i \le N-1$）元素被排除在乘积之外（如图 2-13 所示）。

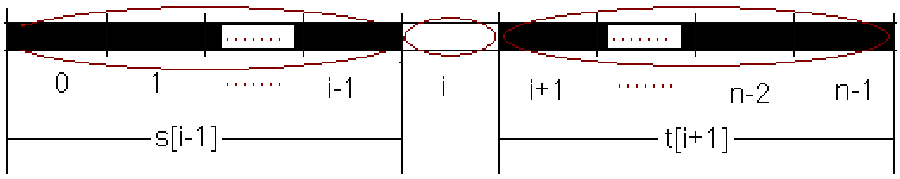

图 2-13　组合示意图

设 *array*[]为初始数组，$s[i]$表示数组前 i 个元素的乘积$s[i]=\prod_{1}^{i} array[i-1]$，其中 $1\le i\le N$，$s[0] = 1$（边界条件），那么 $s[i]= s[i-1]\times array[i-1]$，其中 $i = 1, 2, \cdots, N-1, N$；

设 $t[i]$表示数组后（$N-i$）个元素的乘积$t[i]=\prod_{i}^{n} array[i]$，其中 $1\le i\le N$，$t[N+1]=1$（边界条件），那么 $t[i]=t[i+1]\times array[i]$，其中 $i=1, 2, \cdots, N-1, N$；

那么设 $p[i]$为数组除第 i 个元素外,其他 $N-1$ 个元素的乘积，即有：

$p[i]=s[i-1]\times t[i+1]$。

由于只需要从头至尾，和从尾至头扫描数组两次即可得到数组 s[]和 t[]，进而线性时间可以得到 p[]。所以，很容易就可以得到 p[]的最大值（只需遍历 p[]一次）。总的时间复杂度等于计算数组 s[]、t[]、p[]的时间复杂度加上查找 p[]最大值的时间复杂度等于 O（N）。

【解法二】

其实，还可以通过分析，进一步减少解答问题的计算量。假设 N 个整数的乘积为 P，针对 P 的正负性进行如下分析（其中，A_{N-1}表示 $N-1$ 个数的组合，P_{N-1}表示 $N-1$ 个数的组合的乘积）：

1. P为0

   那么，数组中至少包含有一个0。假设除去一个0之外，其他$N-1$个数的乘积为Q，根据Q的正负性进行讨论：

   Q为0

   说明数组中至少有两个0，那么$N-1$个数的乘积只能为0，返回0；

   Q为正数

   返回Q，因为如果以0替换此时A_{N-1}中的任一个数，所得到的P_{N-1}为0，必然小于Q；

   Q为负数

   如果以0替换此时A_{N-1}中的任一个数，所得到的P_{N-1}为0，大于Q，乘积最大值为0。

2. P为负数

   根据“负负得正”的乘法性质，自然想到从N个整数中去掉一个负数，使得P_{N-1}为一个正数。而要使这个正数最大，这个被去掉的负数的绝对值必须是数组中最小的。我们只需要扫描一遍数组，把绝对值最小的负数给去掉就可以了。

3. P为正数

类似P为负数的情况，应该去掉一个绝对值最小的正数值，这样得到的P_{N-1}就是最大的。

上面的解法采用了直接求 N 个整数的乘积 P，进而判断 P 的正负性的办法，但是直接求乘积在编译环境下往往会有溢出的危险(这也就是本题要求不使用除法的潜在用意☺)，事实上可做一个小的转变，不需要直接求乘积，而是求出数组中正数（+）、负数（-）和 0 的个数，从而判断 P 的正负性，其余部分与以上面的解法相同。

在时间复杂度方面，由于只需要遍历数组一次，在遍历数组的同时就可得到数组中正数（+）、负数（-）和 0 的个数，以及数组中绝对值最小的正数和负数，时间复杂度为 $O(N)$。